Comparative Analysis of CPU Scheduling Algorithms: Insights and Implications

Contents

Introduction

CPU scheduling algorithms play a crucial part in determining the efficiency and fairness of a system. CPU schedulers control which process gets to use the CPU and for how long. Essentially, influencing factors such as response time, throughput, and fairness. Therefore, it is important to understand how different scheduling algorithms behave under varying conditions and what factors impact their performance.

In this report, we conduct an investigation into the performance of five CPU scheduling algorithms: First-Come, First-Served (FCFS), Round Robin (RR), Ideal Shortest Job First (IdealSJF), Multi-level feedback queue with Round Robin (Feedback RR), and Shortest Job First using exponential averaging (SJF).

Our primary objective is to identify the most efficient scheduling algorithm in different scenarios, and also to provide insights by conducting a series of experiments to investigate the performance of the five different scheduling algorithms mentioned above. Specifically, we will be looking at the impact of process characteristics such as burst time and arrival time, as well as the effect of different workload intensities on the performance of each algorithm.

We use a discrete event simulator and an input generator implemented in Java to conduct experiments on different processor loads and schedulers, and measure various performance metrics. To generate input data, we used an input generator that creates a set of processes with varying CPU and I/O burst time and arrival time. We will vary the input parameters to simulate different workload scenarios and analyse the performance of each scheduling algorithm under all of the conditions. Ensuring statistical significance, is done by repeating 5 different random seeds for each experiment for the input generator, and averaged the results of all the processes for each seed. Eventually, we analyse the results to draw meaningful conclusions. This report presents the design and execution of our experiments, our findings, and a discussion of our results.

To visually represent our results, Pandas, a Python library was used to produce graphs and charts that allow for clear and concise analysis, and presentation of our findings.

1 Experiment 1: Comparing Scheduling Algorithms for CPU-intensive Processes

1.1 Methodology

Experiment Objective: To evaluate and compare the performance of different scheduling algorithms for CPU-bound processes under varying workloads.

- Simulator Parameters:
  - timeLimit = 10000
  - periodic = false
  - interruptTime = 0
  - timeQuantum = 50
  - initialBurstEstimate = 35
  - alphaBurstEstimate = 0.5

The timeLimit was set to 10000. As it is long enough to ensure that all processes are finished below the time provided for each scheduler.

Periodic parameters were set to false and interruptTime parameters were set 0 since they are irrelevant in this type of experiment as the focus of the experiment is to compare the performance of different scheduling algorithms for CPU-intensive processes.

The time quantum was established at 100, as it needs to be lengthy enough to prevent excessive overhead from context switching between processes, while simultaneously being brief enough to guarantee equal distribution of CPU time among all processes.

Lastly, in order to prevent overestimation and inaccuracy in burst time predictions, I have set the initial burst estimate to 35 and the alpha burst estimate to 0.5.

- Input Data Parameters:
  - numberOfProcesses = 10
  - staticPriority = 0
  - meanInterArrival = 10.0
  - meanCpuBurst = 100.0
  - meanIOBurst = 5.0
  - meanNumberBursts = 3.0

  First of all, The algorithms were tested on 10 processes, this value allows for a manageable number of processes in the system without being too complex or too simple. It provides a realistic scenario to analyze the scheduling algorithms' performance, and compare their efficiency in handling multiple processes.

  Processes will not be assigned a priority value throughout the experiments since it would not be appropriate for this type of experiments, as it would make all processes have different priority levels. This allows for a fair comparison of the scheduling algorithms without any bias towards high-priority processes. However, algorithms such as Feedback-RR and SJF will assign a priority levels to each process on their own.

  A mean inter-arrival time of 10.0 allows for a steady stream of processes entering the system, which helps to create varying workloads for the scheduler. This will test the scheduling algorithms' adaptability and performance under different workloads.

  Setting the mean CPU burst time to 100.0 creates a CPU-bound environment, as the processes require longer periods of CPU time. This is ideal for testing the performance of scheduling algorithms in a CPU-bound scenario.

  With a lower mean I/O burst time of 5.0, the processes spend less time waiting for I/O operations, emphasizing the importance of efficient CPU scheduling.

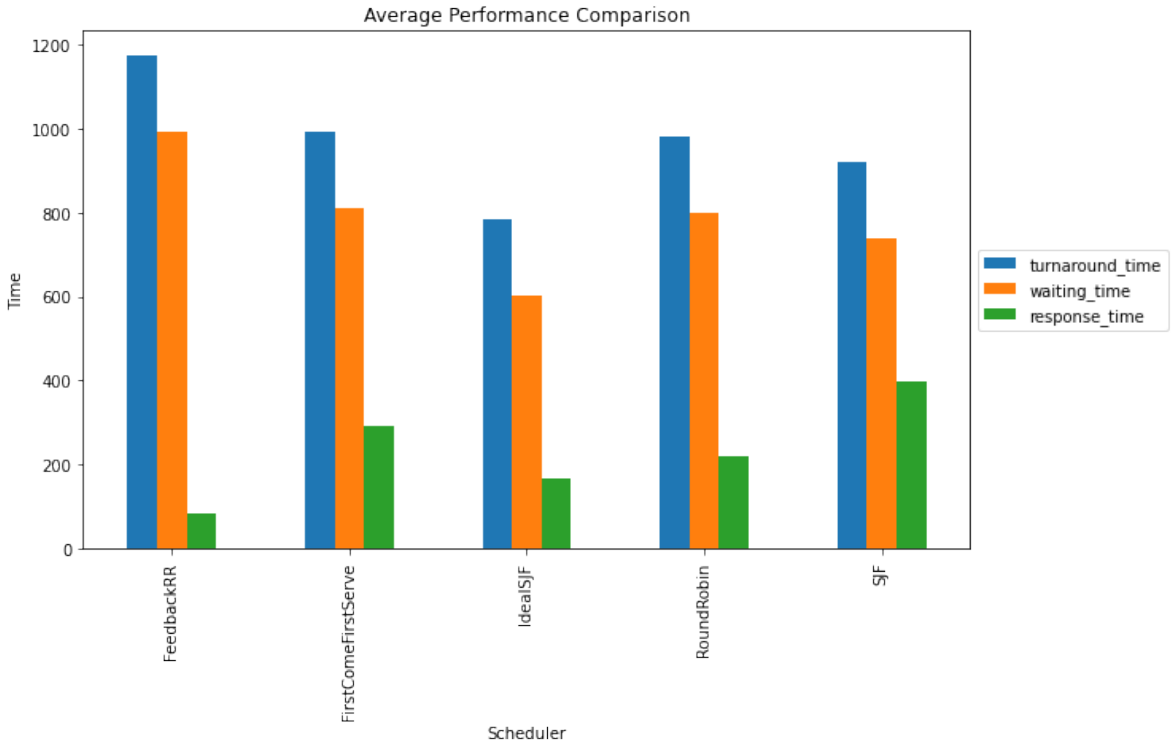
  By setting the mean number of bursts to 3.0, the experiment generates processes with a moderate level of complexity. This allows the scheduling algorithms to be tested with processes that have multiple bursts, which is more representative of real-world scenarios.

We ran the simulator on the input data for each scheduling algorithm and recorded performance metrics such as turnaround time, waiting time, and response time. Lastly, we took the average of these metrics across all runs of the simulator.

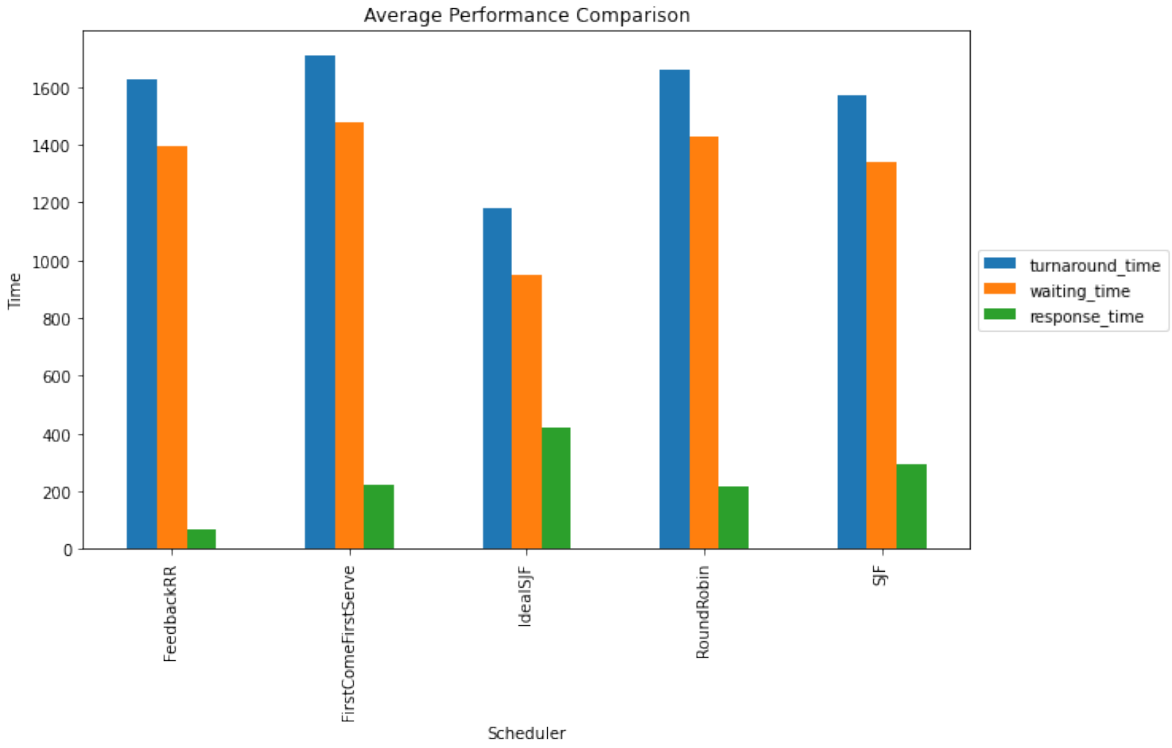
The choice of scheduling algorithm will have a significant impact on the overall performance of a system, depending on the characteristics of the processes being scheduled and the workload being executed. Specifically, we hypothesise that:

When addressing CPU-intensive processes, scheduling algorithms that emphasize shorter job durations, such as IdealSJF and SJF will perform better than algorithms that prioritise fairness (such as FCFS and Round Robin) or responsiveness (such as Feedback-RR).

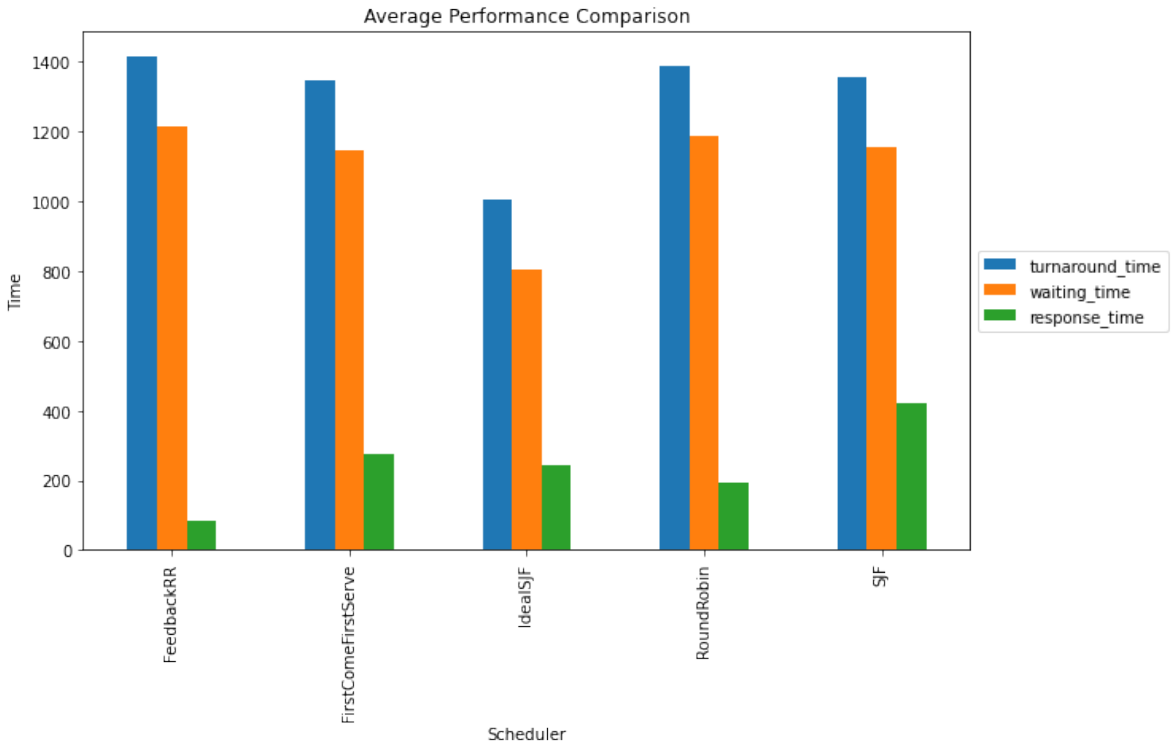
1.2 Results



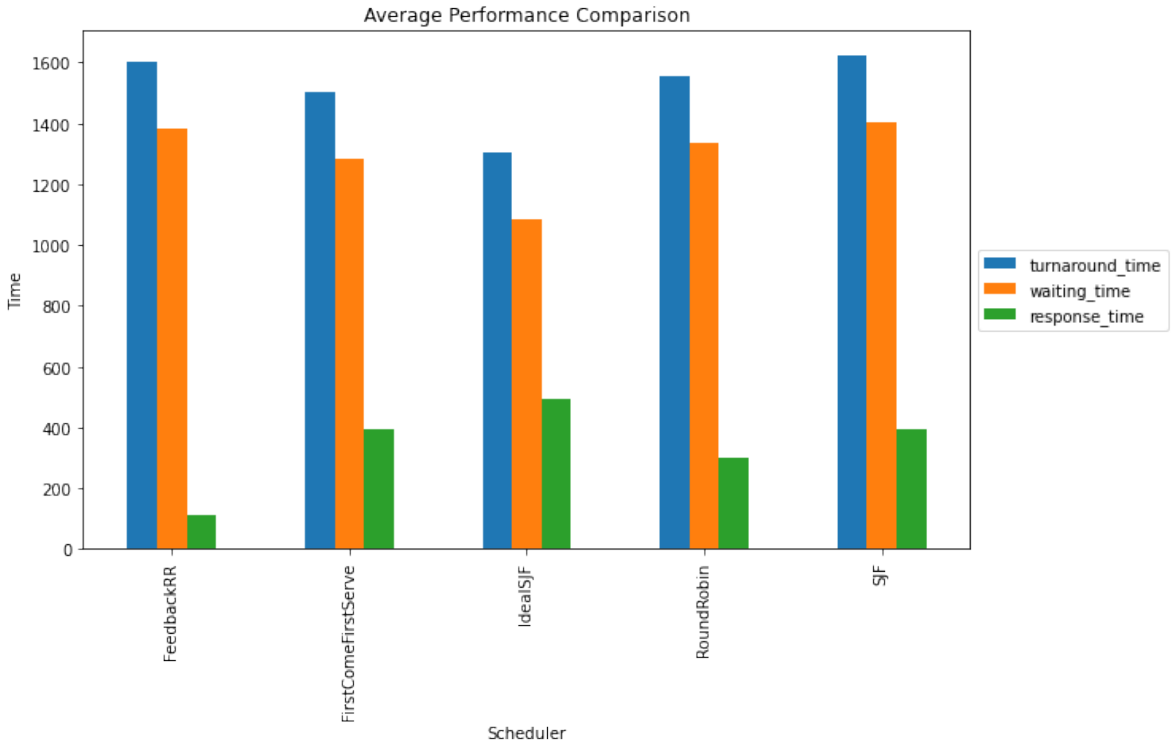
Seed 1, Bar graph comparing the average performance of scheduling algorithms on CPU-intensive processes



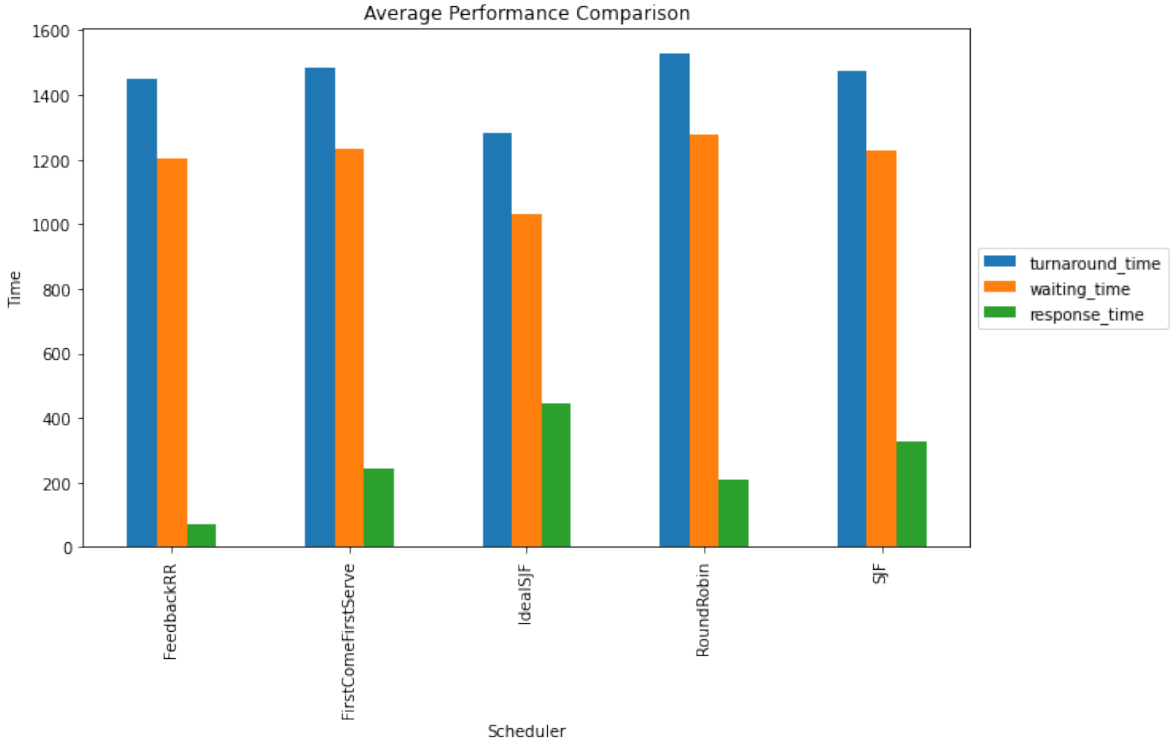
Seed 2, Bar graph comparing the average performance of scheduling algorithms on CPU-intensive processes



Seed 3, Bar graph comparing the average performance of scheduling algorithms on CPU-intensive processes



Seed 4, Bar graph comparing the average performance of scheduling algorithms on CPU-intensive processes



Seed 5, Bar graph comparing the average performance of scheduling algorithms on CPU-intensive processes

1.3 Discussion

Each seed has its own graph, a bar graph, showing a clear visual representation of turnaround time, waiting time, and response time metrics for each scheduling algorithm.

The experimental results provide evidence that both supports and challenges our hypothesis. We will refer to specific graphs and tables in the Results section to support these claims.

Supporting evidence:

1. Turnaround Time: In terms of turnaround time, the results align with our hypothesis. IdealSJF performs the best, followed by SJF (except for one run seed, specifically, in seed 4, where it scored the worst). Feedback-RR, FCFS, and Round Robin scored the worst in this category.

2. Waiting Time: The results for waiting time also support our hypothesis. IdealSJF performs the best, while Feedback-RR and FCFS scored the worst. However, in one run out of five, SJF scored the worst, this is an inconsistency with our hypothesis (in seed 4 graph).

Challenging evidence:

1. Response Time: The results for response time contradict our hypothesis. IdealSJF scored the worst in three out of five seeds (in seed 2, 4 and 5), and SJF scored the worst in the other two runs. Feedback-RR, which prioritizes responsiveness, scored the best out of all algorithms.

In conclusion, our experimental results partially support our hypothesis, as IdealSJF and SJF perform better in turnaround and waiting times. However, the results challenge our hypothesis in terms of response time, where Feedback-RR, an algorithm prioritizing responsiveness, outperforms IdealSJF and SJF. These findings suggest that while scheduling algorithms that emphasize shorter job durations may excel in some performance metrics, they may not be universally superior across all metrics in CPU-intensive processes.

2 Experiment 2: Evaluating Scheduling Algorithms for I/O-intensive Processes

2.1 Methodology

Experiment Objective: To evaluate and compare the performance of different scheduling algorithms for I/O-bound processes under varying workloads.

- Simulator Parameters:
  - timeLimit = 10000
  - periodic = false
  - interruptTime = 0
  - timeQuantum = 10
  - initialBurstEstimate = 10
  - alphaBurstEstimate = 0.5

  The timeLimit was set to 10000. As it is long enough to ensure that all processes are finished below the time provided for each scheduler.

Periodic parameters were set to false and interruptTime parameters were set 0 since they are irrelevant in this type of experiment as the focus of the experiment is to compare the performance of different scheduling algorithms for I/O-intensive processes.

As the CPU bursts are lower in this experiment, the time quantum was established at 10, as it needs to be lengthy enough to prevent excessive overhead from context switching between processes, while simultaneously being brief enough to guarantee equal distribution of CPU time among all processes. Furthermore, it makes algorithms that make use of time quantum (Round Robin and Feedback-RR) fair in this experiment.

Lastly, in order to prevent overestimation and inaccuracy in burst time predictions, I have set the initial burst estimate to 10 and the alpha burst estimate to 0.5.

- Input Data Parameters:
  - numberOfProcesses = 10
  - staticPriority = 0
  - meanInterArrival = 10.0
  - meanCpuBurst = 5.0
  - meanIOBurst = 100.0
  - meanNumberBursts = 3.0

  The algorithms were tested on 10 processes, this value allows for a manageable number of processes in the system without being too complex or too simple. It provides a realistic scenario to analyze the scheduling algorithms' performance, and compare their efficiency in handling multiple processes.

  Processes will not be assigned a priority value throughout the experiments since it would not be appropriate for this type of experiments, as it would make all processes have different priority levels. This allows for a fair comparison of the scheduling algorithms without any bias towards high-priority processes. However, algorithms such as Feedback-RR and SJF will assign a priority levels to each process on their own.

  A mean inter-arrival time of 10.0 allows for a steady stream of processes entering the system, which helps to create varying workloads for the scheduler. This will test the scheduling algorithms' adaptability and performance under different workloads.

  Setting the mean CPU burst time to 5.0 creates an environment where processes require shorter periods of CPU time. This emphasizes the I/O-intensive nature of the processes in this scenario.

  As the focus of the experiment is on I/O-intensive processes, the majority of the time is spent on I/O operations. Thus, the mean I/O burst time of 100.0, the processes spend more time waiting for I/O operations, making them I/O-intensive. This is ideal for testing the performance of scheduling algorithms in an I/O-intensive environment.

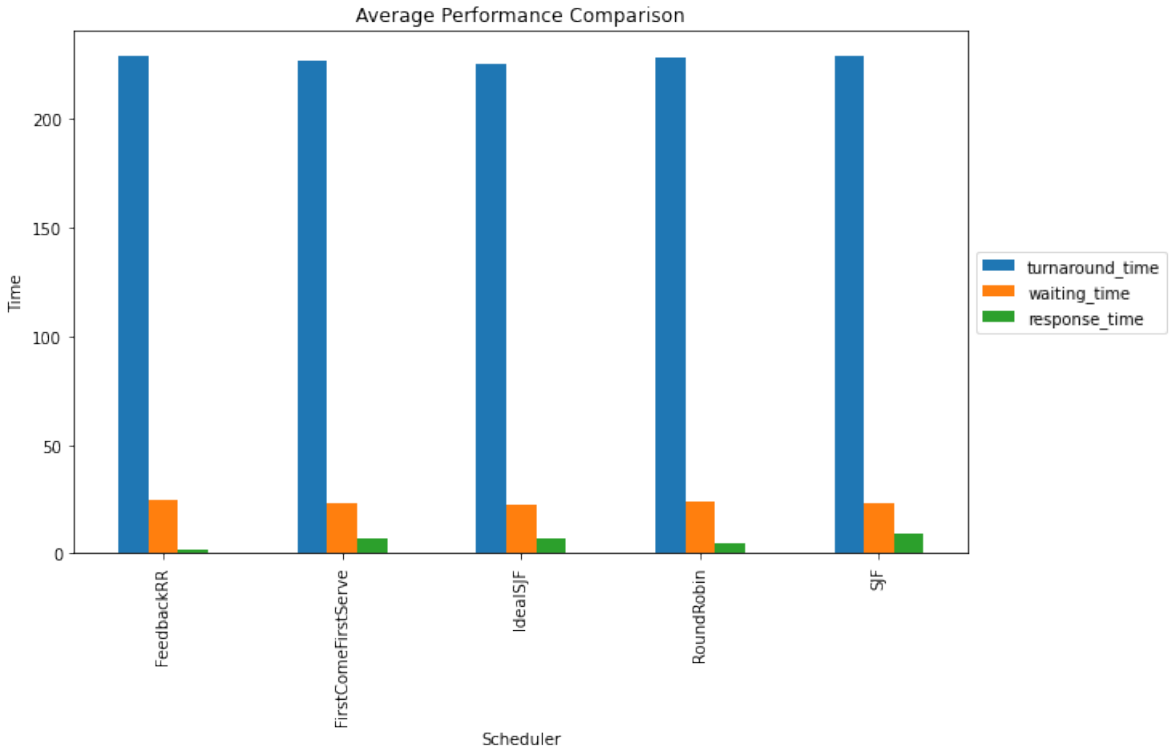
  By setting the mean number of bursts to 3.0, the experiment generates processes with a moderate level of complexity. This allows the scheduling algorithms to be tested with processes that have multiple bursts, which is more representative of real-world scenarios.

We ran the simulator on the input data for each scheduling algorithm and recorded performance metrics such as turnaround time, waiting time, and response time.Lastly, we took the average of these metrics across all runs of the simulator.

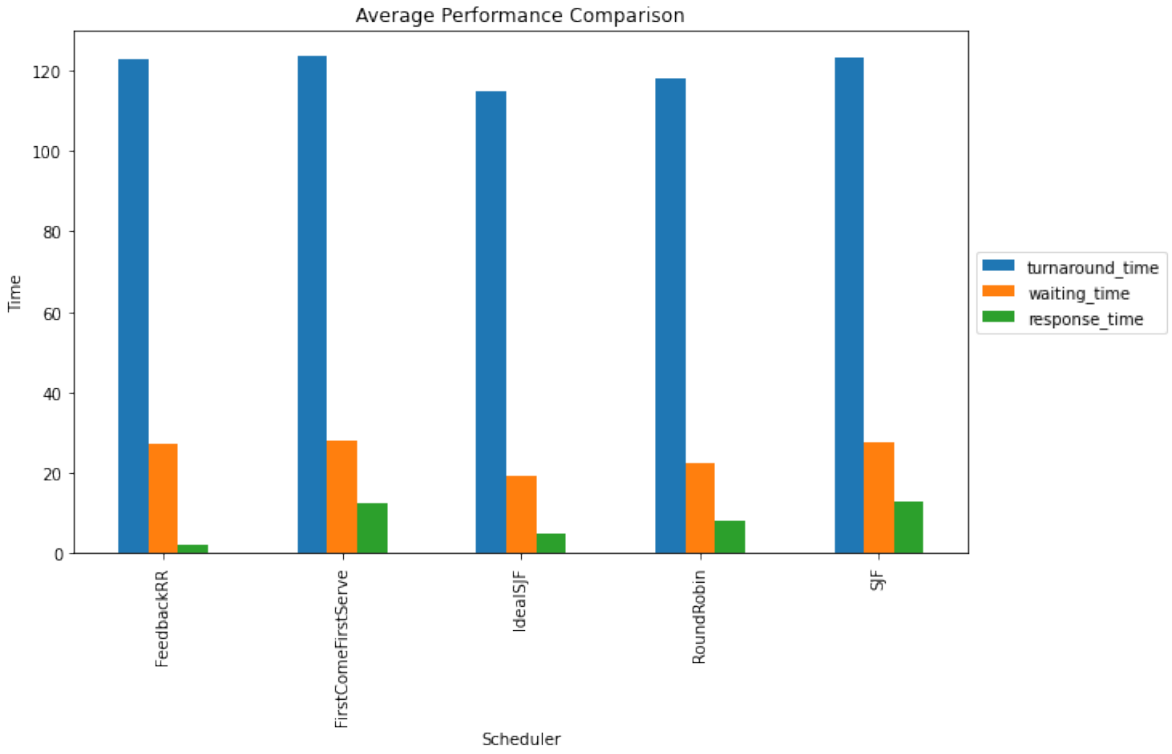
The choice of scheduling algorithm will have a significant impact on the overall performance of a system, depending on the characteristics of the processes being scheduled and the workload being executed. Specifically, we hypothesise that:

For I/O-intensive processes, a scheduling algorithm that allows for efficient use of I/O devices (such as IdealSJF and SJF) or a non-priority-based scheduling algorithm (such as FCFS) will perform better than algorithms that prioritise CPU time (such as Feedback-RR) or fairness (such as Round Robin).

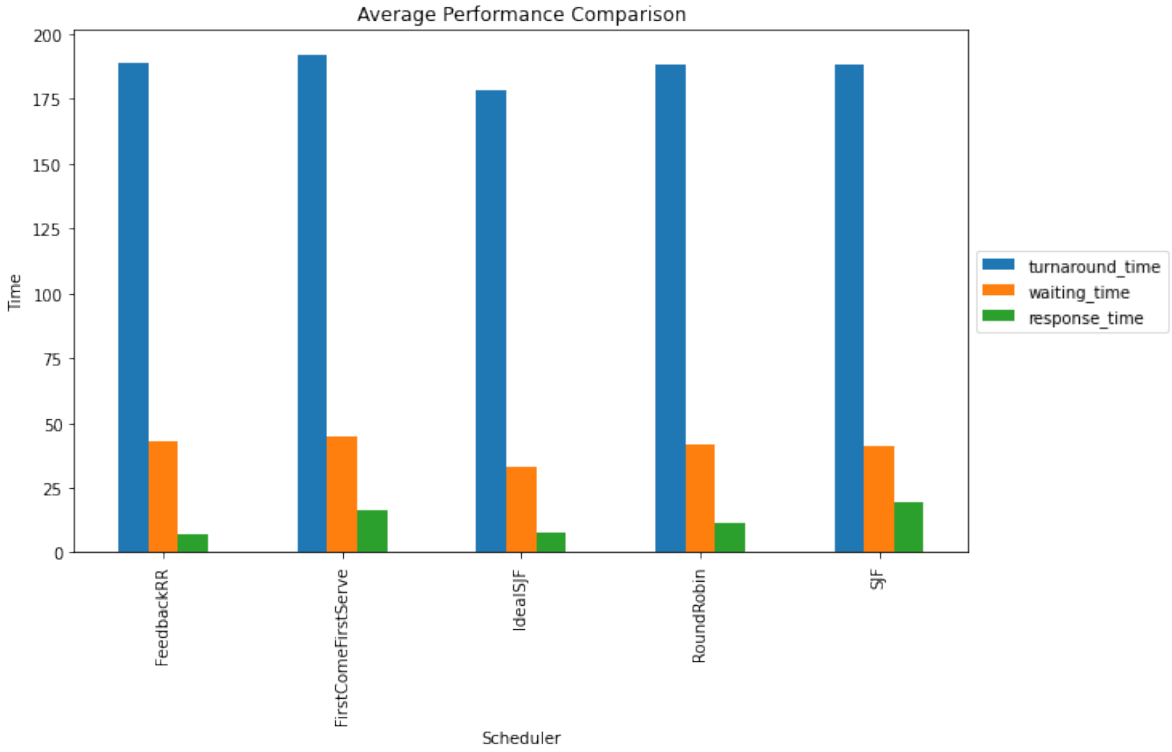
2.2 Results



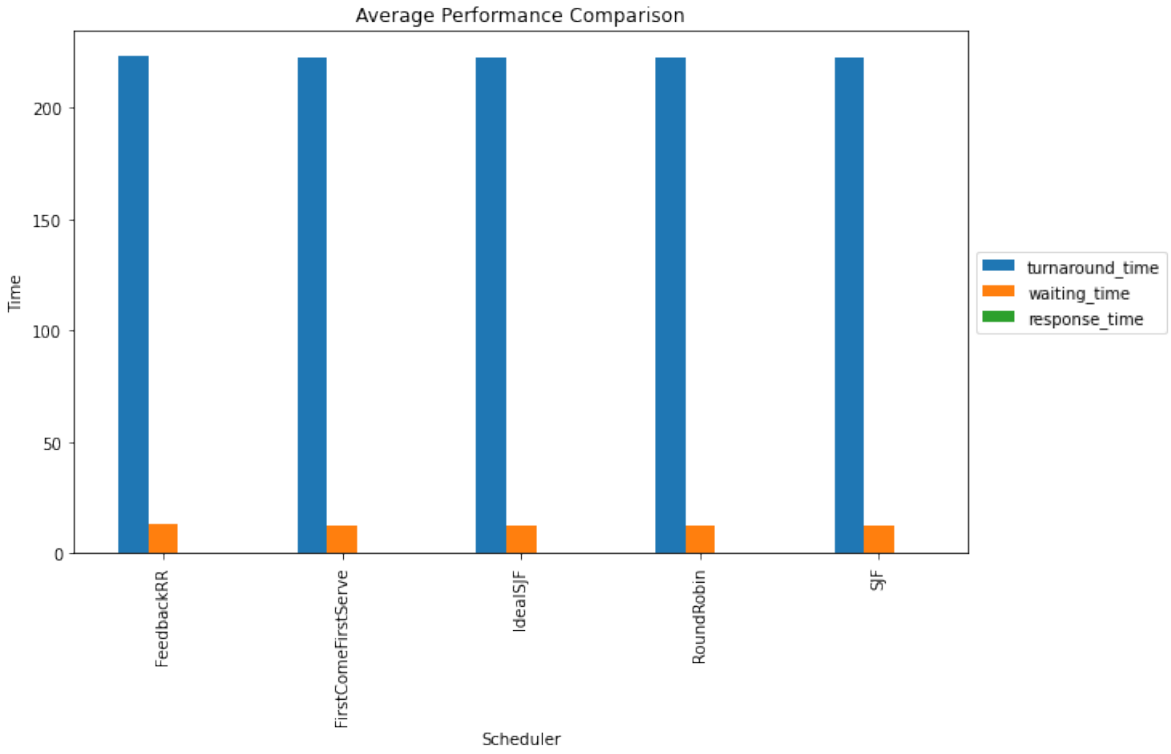
Seed 1, Bar graph comparing the average performance of scheduling algorithms on I/O-intensive processes



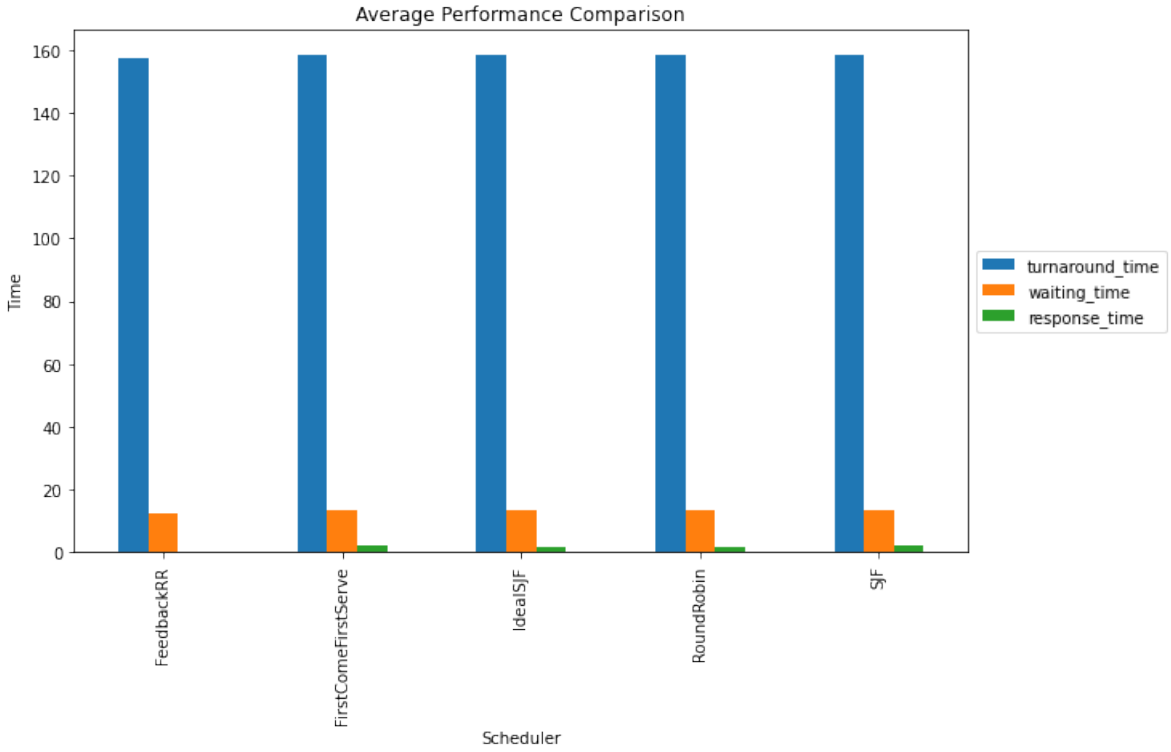
Seed 2, Bar graph comparing the average performance of scheduling algorithms on I/O-intensive processes



Seed 3, Bar graph comparing the average performance of scheduling algorithms on I/O-intensive processes



Seed 4, Bar graph comparing the average performance of scheduling algorithms on I/O-intensive processes



Seed 5, Bar graph comparing the average performance of scheduling algorithms on I/O-intensive processes

2.3 Threats to validity

In experiment 2, the experiment backfired since the turnaround metrics in the graphs were exponentially higher than other two metrics (waiting time and response time), making the graph less visually appealing and harder to interpret. The primary reason for this is because the I/O bursts were high and CPU bursts very low, so naturally, the turnaround time will always be significantly higher.

2.4 Discussion

Similarly, to experiment 1, each seed has its own bar graph, showing a representation of turnaround time, waiting time, and response time metrics for each scheduling algorithm.

The experimental results provide evidence that both supports and challenges our hypothesis. We will refer to specific graphs and tables in the Results section to support these claims.

Supporting evidence:

1. Turnaround Time: IdealSJF performed the best in turnaround time for seeds 1, 2, and 3. This supports our hypothesis as IdealSJF is an algorithm that allows for efficient use of I/O devices.
2. Waiting Time: IdealSJF also performed the best in waiting time for seeds 1, 2, and 3, further supporting our hypothesis.

Challenging evidence:

1. Turnaround Time: In seeds 4 and 5, all algorithms performed nearly the same, and SJF performed the worst in seeds 1 and 2. This contradicts our hypothesis, as we expected SJF and FCFS to perform better.
2. Waiting Time: FCFS performed the worst throughout all the seeds, which contradicts our hypothesis that non-priority-based scheduling algorithms, such as FCFS, would perform better.
3. Response Time: The results for response time contradict our hypothesis. Feedback-RR, an algorithm that prioritizes CPU time, performed the best throughout all the seeds, while FCFS and SJF performed the worst in seeds 1, 2, and 3.

In conclusion, our experimental results partially support our hypothesis, as IdealSJF performs well in both turnaround and waiting times for I/O-intensive processes. However, the results challenge our hypothesis in terms of waiting time for FCFS and response time for Feedback-RR. These findings suggest that while some algorithms may excel in certain performance metrics for I/O-intensive processes, the relationship between the scheduling algorithms and their performance in I/O-intensive processes is more complex than initially hypothesized.

3 Experiment 3: Resilience of Scheduling Algorithms under High Workloads, CPU and I/O intensive Processes

3.1 Methodology

Experiment Objective: To evaluate and compare different scheduling algorithms under high mixed workloads of CPU and I/O intensive processes.

- Simulator Parameters:

- timeLimit = 10000
- periodic = false
- interruptTime = 0
- timeQuantum = 50
- initialBurstEstimate = 35
- alphaBurstEstimate = 0.5

The timeLimit was set to 10000. As it is long enough to ensure that all processes are finished below the time provided for each scheduler.

Periodic parameters were set to false and interruptTime parameters were set 0 since they are irrelevant in this type of experiment as the focus of the experiment is to compare the performance under high mixed workloads of CPU and I/O intensive processes.

The time quantum was established at 100, as it needs to be lengthy enough to prevent excessive overhead from context switching between processes, while simultaneously being brief enough to guarantee equal distribution of CPU time among all processes. Furthermore, it makes algorithms that make use of time quantum (Round Robin and Feedback-RR) fair in this experiment.

Lastly, in order to prevent overestimation and inaccuracy in burst time predictions, I have set the initial burst estimate to 35 and the alpha burst estimate to 0.5.

- Input Data Parameters:
  - numberOfProcesses = 10
  - staticPriority = 0
  - meanInterArrival = 10.0
  - meanCpuBurst = 100.0
  - meanIOBurst = 100.0
  - meanNumberBursts = 4.0

First of all, The algorithms were tested on 10 processes, this value allows for a manageable number of processes in the system without being too complex or too simple. It provides a realistic scenario to analyze the scheduling algorithms' performance, and compare their efficiency in handling multiple processes.

Processes will not be assigned a priority value throughout the experiments since it would not be appropriate for this type of experiments, as it would make all processes have different priority levels. This allows for a fair comparison of the scheduling algorithms without any bias towards high-priority processes. However, algorithms such as Feedback-RR and SJF will assign a priority levels to each process on their own.

A mean inter-arrival time of 10.0 allows for a steady stream of processes entering the system, which helps to create varying workloads for the scheduler. This will test the scheduling algorithms' adaptability and performance under different workloads.

Setting the mean CPU burst time to 100.0 creates an environment where processes require a significant amount of CPU time, making them CPU-intensive. This value contributes to the mixed workload of CPU and I/O intensive processes.

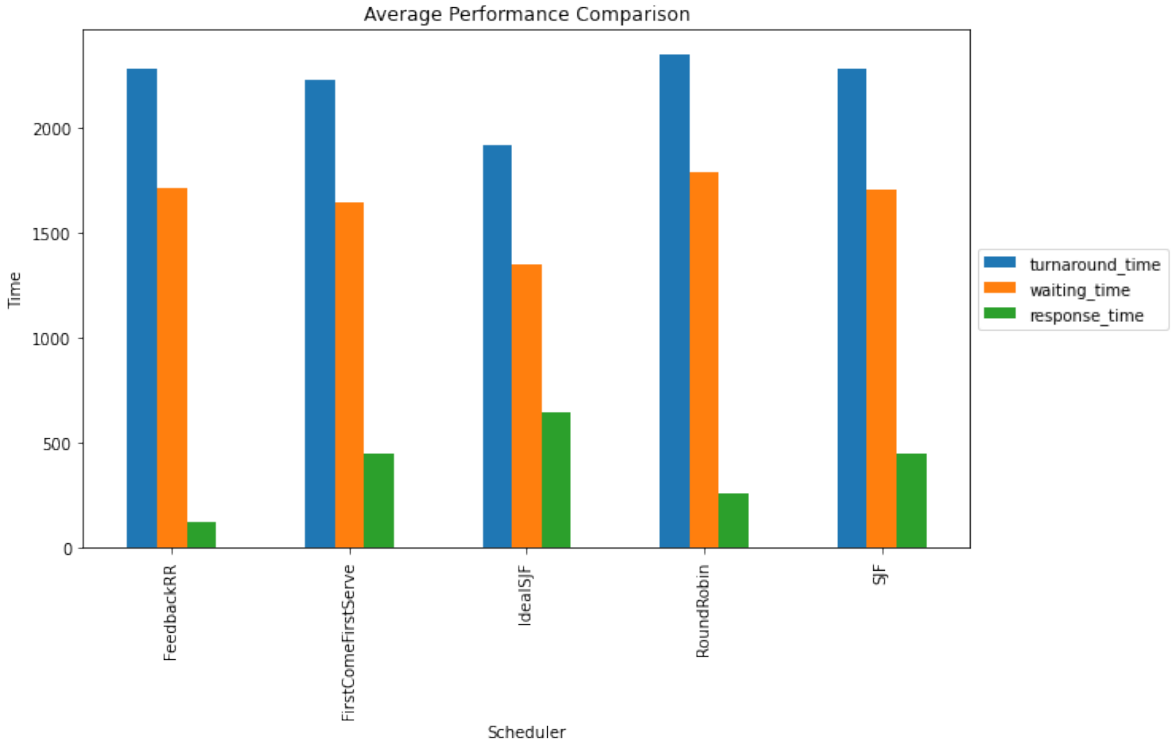
With a mean I/O burst time of 100.0, the processes spend a significant amount of time waiting for I/O operations, making them I/O-intensive. This value also contributes to the mixed workload of CPU and I/O intensive processes.

By setting the mean number of bursts to 4.0 it makes it fair for an experiment that aims to evaluate and compare different scheduling algorithms under high mixed workloads of CPU and I/O intensive processes, as it ensures an even distribution of CPU and I/O bursts. This way, we usually have 2 bursts that use the CPU and 2 bursts that involve I/O. By creating an even mix of bursts like this, we make sure that every scheduling method is tested fairly, with equal chances to perform well with both types of bursts.

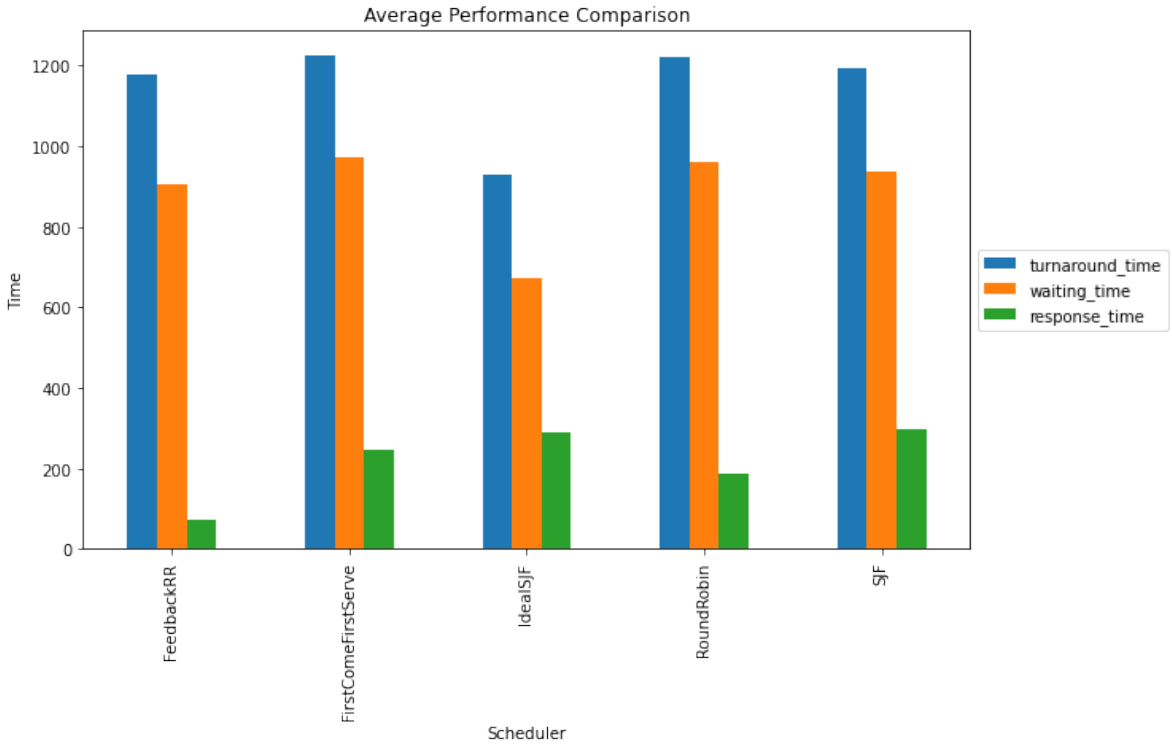
The choice of scheduling algorithm will have a significant impact on the overall performance of a system, depending on the characteristics of the processes being scheduled and the workload being executed. Specifically, we hypothesise that:

For a mixed workload of CPU and I/O-intensive processes, a scheduling algorithm that dynamically adjusts its priority based on the current workload (such as Feedback-RR) will perform better than algorithms that use a fixed priority scheme (such as FCFS, Round Robin and IdealSJF) or algorithms that prioritise specific metrics (such as SJF).

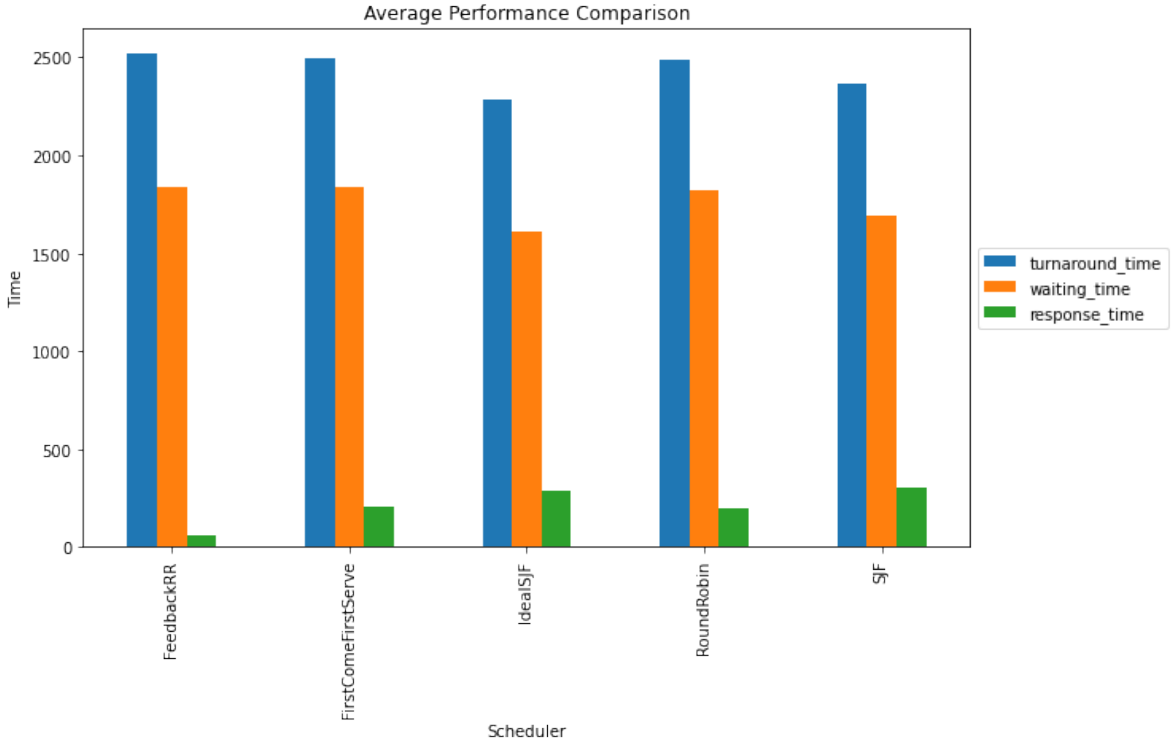
3.2 Results



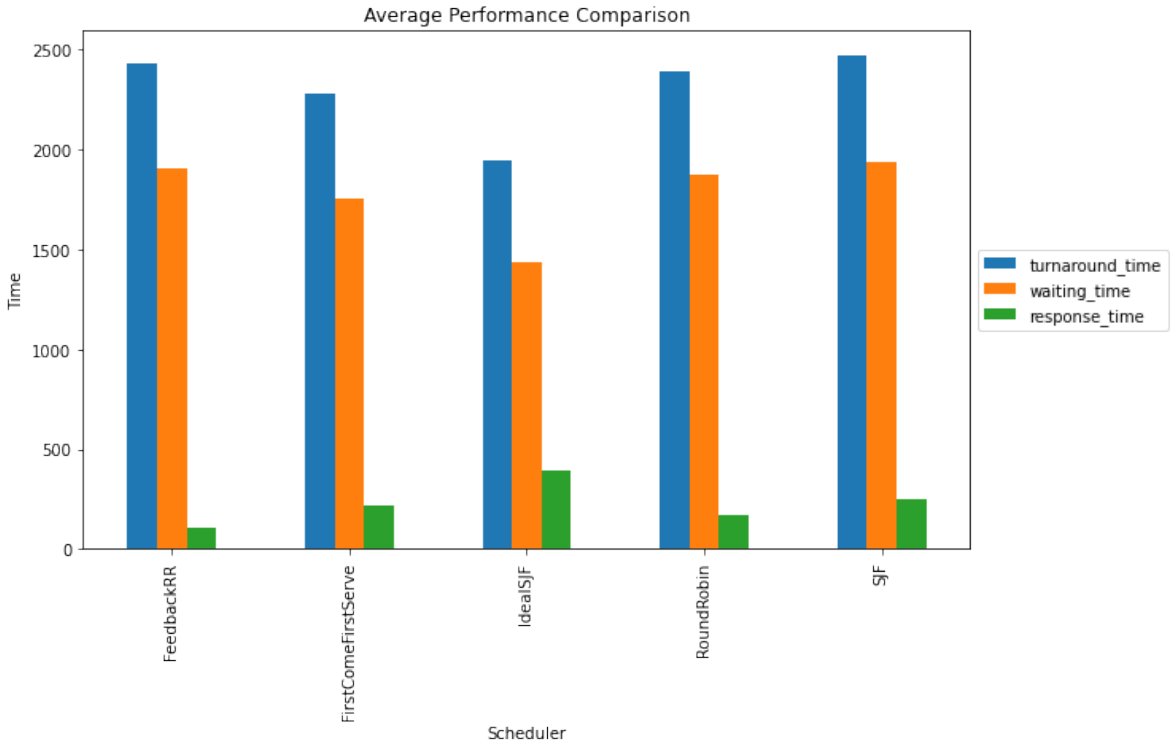
Seed 1, Bar graph comparing the average performance of scheduling algorithms on CPU and I/O intensive processes



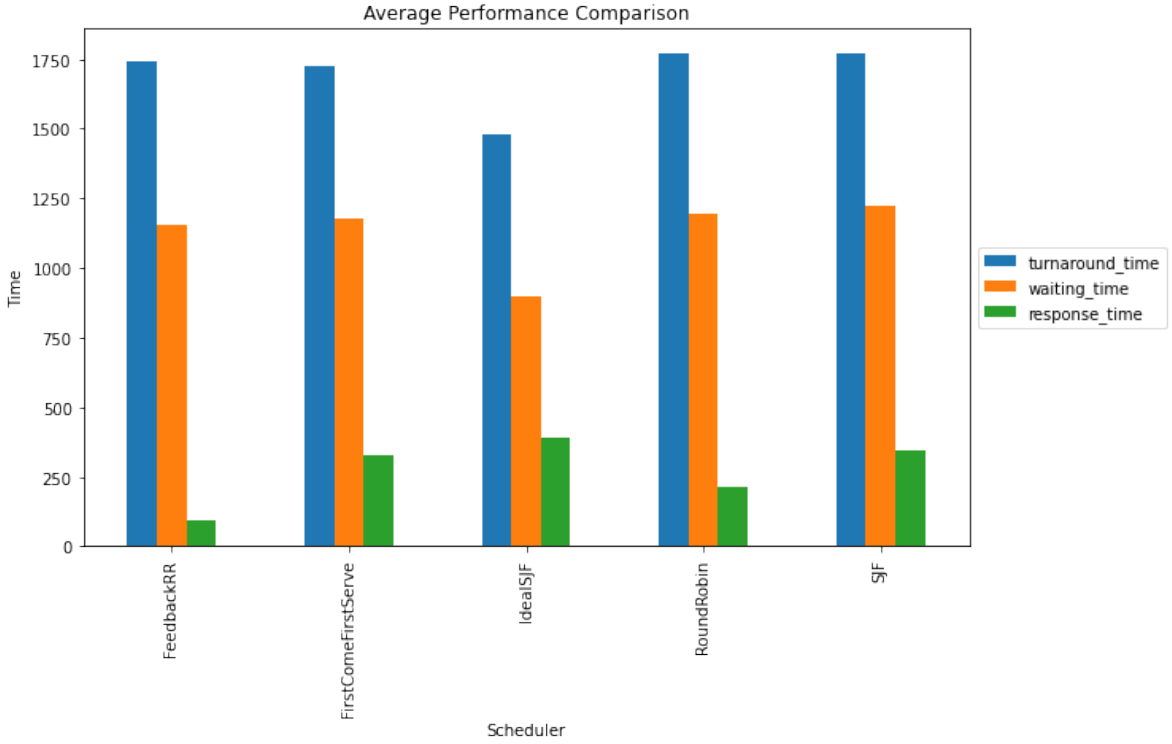
Seed 2, Bar graph comparing the average performance of scheduling algorithms on CPU and I/O intensive processes



Seed 3, Bar graph comparing the average performance of scheduling algorithms on CPU and I/O intensive processes



Seed 4, Bar graph comparing the average performance of scheduling algorithms on CPU and I/O intensive processes



Seed 5, Bar graph comparing the average performance of scheduling algorithms on CPU and I/O intensive processes

3.3 Discussion

Alike the previous experiments, each seed has its own bar graph with the same measurements, being, turnaround time, waiting time, and response time for each scheduling algorithm.

The experimental results provide evidence that both supports and challenges our hypothesis. We will refer to specific graphs and tables in the Results section to support these claims.

Supporting evidence:

1. Response Time: The results for response time support our hypothesis. Feedback-RR, which dynamically adjusts its priority, outperformed all other schedulers throughout all the seeds.

Challenging evidence:

1. Turnaround Time: In turnaround time, IdealSJF, a fixed priority scheme algorithm, outperformed every single scheduler throughout all the seeds. This contradicts our hypothesis, as we expected Feedback-RR to perform better.

2. Waiting Time: The results for waiting time also challenge our hypothesis. IdealSJF outperformed all other schedulers throughout all the seeds, while Feedback-RR, FCFS, Round Robin, and SJF performed nearly the same and had poor performance compared to IdealSJF.

3. Response Time: Surprisingly, Round Robin, a fixed priority scheme algorithm, performed as the second-best scheduler in terms of response time. This contradicts our hypothesis, as we expected algorithms with fixed priority schemes to perform worse.

In summary, our experimental results partially support our hypothesis, as Feedback-RR performs the best in response time for mixed workloads of CPU and I/O-intensive processes. However, the results challenge our hypothesis in terms of turnaround and waiting times, where IdealSJF, a fixed priority scheme algorithm, outperforms all other schedulers. Additionally, Round Robin's performance in response time further contradicts our hypothesis. These findings suggest that the relationship between the scheduling algorithms, and their performance in mixed workloads, is more complex than initially hypothesized and that a dynamic priority-based algorithm like Feedback-RR may not necessarily perform better than fixed priority scheme algorithms across all metrics.

Conclusions

Firstly, IdealSJF and SJF performed well in turnaround and waiting times for CPU-intensive processes, but Feedback-RR had better response times.

Secondly, IdealSJF excelled in turnaround and waiting times for I/O-intensive processes, but the relationship between scheduling algorithms and their performance in I/O-intensive processes was found to be more complex than initially hypothesized.

Thridly, In mixed workloads, Feedback-RR had the best response time, while IdealSJF outperformed other schedulers in turnaround and waiting times, contradicting the hypothesis that dynamic priority-based algorithms like Feedback-RR would be superior across all metrics.

Overall, these findings emphasize the importance of selecting the appropriate scheduling algorithm, based on the specific performance metrics desired for a particular workload. As, there was no algorithm that proved to be universally superior.